

Adv DevOps Lab Exp 04

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Step 1: Create 3 EC2 instances (1 master and 2 slaves). Select SSH option in the inbound rules. Created a key pair to be used commonly between all 3 instances created.

I selected AWS Linux as my operating system and enabled **t2 medium** option for kubernetes cluster to run smoothly

Instances (3) Info						
Last updated 16 minutes ago		Connect	Instance state ▼	Actions ▼	Launch instances ▼	
Find Instance by attribute or tag (case-sensitive)				All states ▼	< 1 > ⚙	
<input type="checkbox"/>	Name ↗	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status
<input type="checkbox"/>	Master	i-0589bc6d5e452907c	Running	t2.medium	2/2 checks passed	View alarms +
<input type="checkbox"/>	Slave2	i-00856c9b137f44cf6	Running	t2.medium	2/2 checks passed	View alarms +
<input type="checkbox"/>	Slave1	i-0beaf1d570ab06d73	Running	t2.medium	2/2 checks passed	View alarms +

Step 2: Open git bash. Change your directory to Downloads and run chmod command on the key pair file that we created for all the EC2 instances that we launched in the earlier step. Hers, i have created a key pair with the name “ec2user” and gave it an extension .pem.

After assigning the key pair file the appropriate permissions, run the ssh command in the following format:

ssh -i <key pair.pem> ubuntu@<public ip address of the instance(DNS)>

Perform this command on all the 3 instances

```

 Dell@DESKTOP-PSTUV9S MINGW64 ~ (main)
 $ cd Downloads

 Dell@DESKTOP-PSTUV9S MINGW64 ~/Downloads (main)
 $ chmod 400 "ec2user.pem"

```

```

 Dell@DESKTOP-PSTUV9S MINGW64 ~/Downloads (main)
 $ ssh -i ec2user.pem ubuntu@ec2-35-175-113-217.compute-1.amazonaws.com
 The authenticity of host 'ec2-35-175-113-217.compute-1.amazonaws.com (35.175.113.217)' can't be established.
 ED25519 key fingerprint is SHA256:ahGnOA8a2dwmhd/8lhRe2a2C6tyvwt2t0bNhpR5/PjM.
 This key is not known by any other names.
 Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
 Warning: Permanently added 'ec2-35-175-113-217.compute-1.amazonaws.com' (ED25519)
 to the list of known hosts.
 Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

 System information as of Sat Sep 14 11:31:16 UTC 2024
 System load: 0.0          Processes: 104
 Usage of /: 22.8% of 6.71GB Users logged in: 0
 Memory usage: 19%        IPv4 address for enx0: 172.31.71.82
 Swap usage: 0%

 Expanded Security Maintenance for Applications is not enabled.
 0 updates can be applied immediately.
 Enable ESM Apps to receive additional future security updates.
 See https://ubuntu.com/esm or run: sudo pro status

 The list of available updates is more than a week old.
 To check for new updates run: sudo apt update

 The programs included with the Ubuntu system are free software;
 the exact distribution terms for each program are described in the
 individual files in /usr/share/doc/*/copyright.

 Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
 applicable law.

 To run a command as administrator (user "root"), use "sudo <command>".
 See "man sudo_root" for details.

```

Step 3: Docker Installation

Perform this step on all 3 instances

```
[ec2-user@ip-172-31-22-31 ~]$ yum install docker -y
Error: This command has to be run with superuser privileges (under the root user on most systems).
[ec2-user@ip-172-31-22-31 ~]$ sudo su
[root@ip-172-31-22-31 ec2-user]# yum install docker -y
Last metadata expiration check: 0:03:24 ago on Sat Sep 14 14:54:57 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
docker	x86_64	25.0.6-1.amzn2023.0.2	amazonlinux	44 M
Installing dependencies:				
containerd	x86_64	1.7.20-1.amzn2023.0.1	amazonlinux	35 M
iptables-libs	x86_64	1.8.8-3.amzn2023.0.2	amazonlinux	401 k
iptables-nft	x86_64	1.8.8-3.amzn2023.0.2	amazonlinux	183 k
libcgroup	x86_64	3.0-1.amzn2023.0.1	amazonlinux	75 k
libnetfilter_conntrack	x86_64	1.0.8-2.amzn2023.0.2	amazonlinux	58 k
libnftnl	x86_64	1.0.1-19.amzn2023.0.2	amazonlinux	30 k
libnftnl	x86_64	1.2.2-2.amzn2023.0.2	amazonlinux	84 k
pigz	x86_64	2.5-1.amzn2023.0.3	amazonlinux	83 k
runsc	x86_64	1.1.13-1.amzn2023.0.1	amazonlinux	3.2 M

```
Transaction Summary
Install 10 Packages
Total download size: 84 M
Installed size: 317 M
```

Next, we are supposed to configure cgroup in a daemon.json file.

Run the following commands

```
cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
```

sudo systemctl enable docker

sudo systemctl daemon-reload

sudo systemctl restart docker

```
[root@ip-172-31-21-0 ec2-user]# cd /etc/docker
[root@ip-172-31-21-0 docker]# cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
[root@ip-172-31-21-0 docker]# sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@ip-172-31-21-0 docker]# sudo systemctl daemon-reload
[root@ip-172-31-21-0 docker]# sudo systemctl restart docker
[root@ip-172-31-21-0 docker]# # Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@ip-172-31-21-0 docker]# # This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core/stable/v1.31/rpm/
enabled=1
```

Step 4: Kubernetes Installation

Carry out this step on all 3 instances

Now, there are a number of steps to install kubernetes onto our instances

1. Set SELinux to permissive mode. These instructions are for Kubernetes 1.31.

Set SELinux in permissive mode (effectively disabling it)

```
sudo setenforce 0
```

```
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

2. Add the Kubernetes yum repository. The exclude parameter in the repository definition ensures that the packages related to Kubernetes are not upgraded upon running yum update as there's a special procedure that must be followed for upgrading Kubernetes. Please note that this repository have packages only for Kubernetes 1.31; for other Kubernetes minor versions, you need to change the Kubernetes minor version in the URL to match your desired minor version (you should also check that you are reading the documentation for the version of Kubernetes that you plan to install).

This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
```

```
[kubernetes]
```

```
name=Kubernetes
```

```
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
```

```
enabled=1
```

```
gpgcheck=1
```

```
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
```

```
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
```

```
EOF
```

3. Install kubelet, kubeadm and kubectl:

```
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
```

4. (Optional) Enable the kubelet service before running kubeadm:

```
sudo systemctl enable --now kubelet
```

```
[root@ip-172-31-22-31 ec2-user]# # Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@ip-172-31-22-31 ec2-user]# # This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@ip-172-31-22-31 ec2-user]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes                                         48 kB/s | 9.4 kB    00:00
Dependencies resolved.
=====
Package                               Architecture      Version           Repository      Size
=====
Installing:
kubeadm                               x86_64            1.31.1-150500.1.1 kubernetes      11 M
```

```

Installing      : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64      6/9
Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64      6/9
Installing      : kubelet-1.31.1-150500.1.1.x86_64                7/9
Running scriptlet: kubelet-1.31.1-150500.1.1.x86_64                7/9
Installing      : kubeadm-1.31.1-150500.1.1.x86_64                8/9
Installing      : kubect1-1.31.1-150500.1.1.x86_64                9/9
Running scriptlet: kubect1-1.31.1-150500.1.1.x86_64                9/9
Verifying       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64      1/9
Verifying       : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64 2/9
Verifying       : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64 3/9
Verifying       : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64 4/9
Verifying       : cri-tools-1.31.1-150500.1.1.x86_64              5/9
Verifying       : kubeadm-1.31.1-150500.1.1.x86_64                6/9
Verifying       : kubect1-1.31.1-150500.1.1.x86_64                7/9
Verifying       : kubelet-1.31.1-150500.1.1.x86_64                8/9
Verifying       : kubernetes-cni-1.5.1-150500.1.1.x86_64          9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64      cri-tools-1.31.1-150500.1.1.x86_64
  kubeadm-1.31.1-150500.1.1.x86_64                kubect1-1.31.1-150500.1.1.x86_64
  kubelet-1.31.1-150500.1.1.x86_64                kubernetes-cni-1.5.1-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64  libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[root@ip-172-31-22-31 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-22-31 ec2-user]#

```

After installing Kubernetes, we need to configure internet options to allow bridging.

- Sudo swapoff-a
- echo "net.bridge.bridge-nf-call-iptables=1"|sudo tee -a /etc/sysctl.conf
- Sudo sysctl-p

Step 5:

On master machine

Run command ...kubeadm init with the proper network pod, here it is, --pod-network-cidr=10.244.0.0/16 to initialize kubernetes

```

[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubect1 apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.22.31:6443 --token plmkzy.2ocxer4410uwlqk4 \
--discovery-token-ca-cert-hash sha256:2590fd7ba571e7e92b4f18f77c2149583f19f6049e3dfb4d306ac22cf2f465d6
[root@ip-172-31-22-31 ec2-user]#

```

We are supposed to add a networking plugin named flannel with the help of the command mentioned in the console output

i.e

kubect1 apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

```

[root@ip-172-31-29-225 ec2-user]# kubect1 apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel
.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created

```

Our cluster is up and running

Now, we need to deploy nginx server. For that purpose, use one of these commands

1. `kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml`
2. `kubectl apply -f https://k8s.io/examples/application/deployment.yaml`

Unfortunately, I was not able to execute this command on my master instance after initializing kubernetes. This is the output that was displayed to me in the console.

```
[root@ip-172-31-29-225 ec2-user]# kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml
error: error validating "https://k8s.io/examples/pods/simple-pod.yaml": error validating data: failed to download openapi: Get "https://172.31.29.225:6443/openapi/v2?timeout=32s": dial tcp 172.31.29.225:6443: connect: connection refused; if you choose to ignore these errors, turn validation off with --validate=false
[root@ip-172-31-29-225 ec2-user]#
```

To resolve this issue, I tried typing the command manually. I even tried freeing up port 22 by running certain commands mentioned in chatgpt. Even then the issue was not resolved.

Hereafter i have attached the images that lead to successful execution of the experiment

- 1) Now that the cluster is set, apply the deployment file of nginx using this command

- `kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml`

```
[root@ip-172-31-28-78 docker]# kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml
pod/nginx created
[root@ip-172-31-28-78 docker]#
```

- 2) Use the command

- `kubectl get pods`: To get the list of pods in the cluster.

```
[root@ip-172-31-28-78 docker]# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
nginx         0/1     Pending   0           23s
```

This output shows that the pod is in a 'PENDING' state, to change it to 'RUNNING' state, run the following commands.

- `kubectl describe pod/nginx`: Provides details about your pod. This command is used to get details about the pod and potential issues with the pod

```
[root@ip-172-31-27-25 docker]# kubectl describe pod nginx
Name:          nginx
Namespace:     default
Priority:       0
Service Account: default
Node:          <none>
Labels:        <none>
Annotations:   <none>
Status:        Pending
IP:            <none>
IPs:           <none>
Containers:
  nginx:
    Image:      nginx:1.14.2
    Port:       80/TCP
    Host Port:  0/TCP
    Environment: <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-...
```

```
QoS Class:      BestEffort
Node-Selectors: <none>
Tolerations:    node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason             Age   From                      Message
  ----     -
Warning   FailedScheduling   10s   default-scheduler         0/1 nodes are available: 1 node(s) had untolera
: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.
[root@ip-172-31-27-25 docker]#
```

3) From this output, we get to know that the node has some untoleraed taint. To remove this, use

- kubectl [taintnodes--allnode-role.kubernetes.io/control-plane:NoSchedule](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#taintnodes)

```
[root@ip-172-31-23-234 docker]# kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-23-234.ec2.internal untainted
```

4) Now, we check the status of the pod by running 'kubectl get pods' again

```
[root@ip-172-31-27-25 docker]# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     1/1     Running   0          2m23s
[root@ip-172-31-27-25 docker]#
```

5) Now, change the port to which you want to host your server on using command

- kubectl port-forward nginx :80

```
[root@ip-172-31-23-234 docker]# kubectl port-forward nginx-deployment-77d8468669-s77nc 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

6) To check whether the deployment was successful, run the command

- `curl--head http://127.0.0.1`: If the terminal returns a status code of 200, it means that the deployment is successful.

Conclusion:

In this experiment, we have learned how to deploy an nginx server to a kubernetes cluster. We also learned how to tackle any intolerable taints that tend to give issues while deploying the server. We also learned how to set the port on which you want to host the server